Java中的数组学习

1. **数组**就是**同一类型数据**的集合，其实数组就是个**容器**。数组可以给其中的数组元素进行编号，方便操作这些**数组元素**。**数组**一旦定义了，**长度**就不能再发生变化。
2. **Java中给定了一个数组工具类，专门用于处理数组的，存在于java.util包中，即Arrays类，其中的方法主要有二分查找和排序，当然还可以利用asList方法把数组转换成集合进行操作。如ArrayList就是利用数组结构实现的集合。**
3. 数组：相同数据类型数据的集合。
4. 如何定义一个数组（主要是两种①和②）：
5. 数据类型[] 数组名字 = new 数据类型[元素个数]；（推荐使用）

如int[] a = new int[10];仅仅只是定义一个数组。在不明确数据的前提下，使用这种方法定义。

②数据类型[] 数组名字 = new 数据类型[]{元素1，元素2，……..}；

例如int[] a = new int[]{1,2,3,4}; 定义并初始化一个数组，数组长度根据元素个数指定，不能再显式地指定。

③ 第三种定义方法：

数据类型[] 数组名字 = {元素1，元素2，……..}；这种方法与第二种方法差不多，但是有些细微的差别，因为new本身就是一种运算。

第二种和第三种方法在知道全部数据的时候使用。

④（不建议） 数据类型 数组名字[] = new int[10];(不建议使用)

注意：前面中括号中间是空的，不要加数字，否则就错了，元素个数子在后面中括号中给出来。③ 数据类型[] 数组名字 = new 数据类型[]{初始化列表}

(2) 如何引用数组元素，如a[0]表示第一个元素；注意：数据元素的索引从0开始的，这点和C语言一样；

（3） 赋值方式： ① int[] a = {1,2,3,2};

② int[] a = new int[]{1,2,3,4};注意中括号中都不要加上数字，因为元素个数就指明了个数问题。直接在后面加上大括号，初值放在大括号里面即可。

（4） **java中的每个数组都有一个名为length的属性，表示数组的长度，**例如 a.length就是个数值。Length属性是public、final、int的，不可以对此赋值，数组一旦确定，就不能更改大小了。

1. 输出语句中直接输出数组名（一维和二维都是一样的），会打印一个字符串，@左边代表**实体的类名**，@右边代表**实体的哈希值**。
2. **二维数组**：**定义方式：int[][] arr = new int[4][12];（java中建议的定义方式，每个小数组的长度相同情况）第二种定义方式：int[][] arr = new int[3][];第二维的长度可以不指定，因为其长度可以不相同。可以对每个二维数组进行初始化，且长度可以不等长，例如**

**arr[0] = new int[4]; arr[1] = int[4]; arr[2] = new int[5]。**

**int[] arr[] = new int [4][12];这样定义也是正确的。**

定义并赋予初值： int[][] a = new int[][]{{1,2,3},{2,3,4},{3,4,5}};

引申：多维数组。三维数组 int[][][] aaa = new int[2][3][4];

1. **求数组的长度方法**：
2. 求一维数组长度 int[] arr = new int[10]; arr.length 即可以求出数组a的长度；
3. 求二维数组长度： int[][] arr = new int[5][10];

arr.length 求的是有几个小数组（有几行），返回5；

a[1].length 既可以求出第二维的长度，返回10。

1. **二维数组的遍历**： 二维数组中的每个小数组的长度可以不同。

实例：**int**[][] arr = **new** **int**[3][];

System.***out***.println(arr.length);

arr[0] = **new** **int**[10];

arr[1] = **new** **int**[8];

arr[2] = **new** **int**[4];

**for**(**int** i = 0;i < arr.length;i++) {

System.***out***.println(i+".length = "+arr[i].length);

}

**for**(**int** i = 0;i < arr.length;i++) {

**for**(**int** j = 0;j < arr[i].length;j++) {

arr[i][j] = i\*j;

System.***out***.print(arr[i][j]+ "\t");

}

System.***out***.println();

}

1. System.out.println(\*\*\*\*); 与 System.out.print(\*\*\*); 的区别：

前者是**先输出后换行**，后者**只是输出**。前者相当于执行后者后加上一句**System.out.println();** 直接换行。

1. C语言中存在指针的概念，而java中不存在指针的概念。
2. 对数组的基本操作就是**存**和**取**。
3. **获取最值问题：从数组中找出最值来：方法是打擂台法。具体又分为两种：**
4. **一是以数组元素的值为记录标准（需要注意给max或min需要赋予数组中的一个元素值，否则默认0会参加比较，可能带来错误），**
5. **二是利用数组元素的角标为记录标准（此时不用给max或min赋予数组元素值）。因为数组中元素值与元素索引是一一对应的。**
6. **排序问题：**（**选择排序**和**冒泡排序**）

首先选择冒泡法：两次循环，第一次循环用来表示趟数（总数减一）；第二次循环表示第i趟需要进行多少次相邻两个数的比较。（总数-1-i）其循环内部用到的就是相邻两个数据的不断判断和交换。

冒泡法的优化：在每一趟中记录两个交换的个数或者利用布尔量，发生任何一个交换就是false，否则为true，从而得知是否已排好，如果在一趟中没有发生任何两个交换，则就意味着已经是拍好顺序，后面的几趟就不用比较了。

1. （重点）**排序法**（共有七种）分类：（一）交换排序：**冒泡法**

（二）快速排序：（其中也存在数据交换）可以使用递归的方法来

实现，找到一个数，然后大于它的放在右边，小于它的放在左边。

（三）**插入排序**：

（四）**选择排序**：（以从小到大为例）就是让第一个元素与后面的n-1个元素依次比较大小，如果第一个元素较大，则发生交换，最终是第一个元素的值就是整个数组中最小的值，然后再利用第二个元素与后面的元素进行比较即可。

（五）**归并排序**

1. 其实在实际开发中，开发人员没有必要进行手写数组排序程序，因为JDK中已经提供了数组的排序方法。**Arrays.sort（数组名）；直接调用即可。**
2. **二分查找（Binary Search）：需要数组中的数据是有序的。**
3. 数组索引超出范围异常：ArrayIndexOutOfBoundsException。当访问到数组中不存在的角标时，就会发生此异常。
4. 空指针异常：NullPointerException。当引用型变量没有指向任何实体（即为null值）时，如果操作此实体就会出现空指针异常。
5. **选择法排序：**

**private** **static** **void** selectSort(**int**[] arr)

{

**int** temp;

**for**(**int** i = 0;i < arr.length-1;i++) {

**for**(**int** j = i + 1;j < arr.length;j++) {

**if**(arr[i] > arr[j]) {

**swap(arr,i,j)；//见下面的函数，一定要把数组穿进去**

}

}

}

}

1. **冒泡法排序：**

private static void bubbleSort(int[] arr)

{

int temp;

for(int i = 0;i < arr.length-1;i++) {

for(int j = 0;j < arr.length - i - 1;j++) {

if(arr[j] > arr[j + 1]) {

swap(arr,j,j+1);//见下面的函数

}

}

}

}

**private** **static** **void** swap(**int**[] arr, **int** i, **int** j)

{

**int** temp;

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

1. **数组排序的性能问题：选择法排序，第一轮，是让第一个元素与后面元素依次比较，然而可能会发生多次数据元素的交换，改进：可以每次比较后，不进行数组元素的交换，而是仅仅记录下相应元素的索引等到一轮结束后，再进行数组元素的交换，这样一轮也就需要一次数组元素交换即可。**

**改进：private** **static** **void** selectSortAdvanced(**int**[] arr)

{

**for**(**int** i = 0;i < arr.length-1;i++) {

**int** num = arr[i];

**int** index = i;

**for**(**int** j = i + 1;j < arr.length;j++) {

**if**(num > arr[j]) {

num = arr[j];

index = j;

}

**if**(index != i)

*swap*(arr,i,index);

}

}

}

1. **数组中的查找技术：折半查找（二分查找）**

**普通方法：** **private** **static** **int** getIndex(**int**[] arr, **int** key)

{

**for**(**int** i = 0;i < arr.length;i++) {

**if**(arr[i] == key) {

**return** i;

}

}

**return** -1;

}

**折半查找（二分查找binarySearch）：适用于已经排好顺序的数组序列从大到小或从小到大均可。**

**private** **static** **int** binarySearch(**int**[] arr, **int** key)

//前提：arr一定是排好顺序的数组。

{

**int** min,max,mid;

max = arr.length-1;

min = 0;

**if**(key == arr[min])

**return** min;

**if**(key == arr[max])

**return** max;

**while**(max >= min) {

mid = (min + max)/2;

//除以2，相当于右移一位即 mid =（min + max）>>1; 注意位操作是非常快的

**if**(key == arr[mid])

**return** mid;

**else** **if**(key > arr[mid]) {

min = mid +1;

}

**else**

max = mid - 1;

}

**return** -1;

}

1. **关于二分查找的一个面试题目**： 给定一个有序的数组，如果往该数组中存储一个元素，并保证这个数组还是有序的，怎么获取这个元素应该存放的角标。

答 ： 利用JDK提供的Arrays.binarySearch（）方法即可实现。如果数组中存在查找的的数据，返回的就是此数值的角标，如果不存在此数值，返回的就是一个负数，其绝对值是此数值应当插入的第几个位置处（也就是索引加1）.在刚才的基础上把-1换成min就可以了。

解析：如果要存放的元素，在数组中存在，则直接找到此元素的索引即可，但是如果不存在此元素，那么最后肯定会是min 和 max值是紧挨着的，下一步就是min 加1，而max减1；且分析知需要存放的数据应该存放到到大于此数值的第一个元素位置，所以返回min。例如：arr[9]=100;arr[10]=200; key = 150; min = 9;max = 10;mid = 9;接下来，由于key 大于arr[mid] 所以，min = mid +1 =10,所以返回min = 10,正确。

1. JDK中已经提供了二分查找方法 Arrays.binarySearch(arr,key);此方法实现的功能是这样的，如果数组中存在可以，直接返回索引，如果不存在此数值，则返回一个负数，且此负数的绝对值，就是说明需要查找数组存到数组第几个位置（不是索引啊，比索引加1）上可以保持数组仍然是有序的。注意：JDK提供的sort（）方法和binarySearch（）方法都是针对升序的ascending order.，不适用于原数组是降序的。
2. **什么时候使用数组呢？（查表法是数组的一种常见的应用。）**

当数据出现了对应关系，而且对应关系的一方是有序的编号，并可以作为角标使用，这时候使用数组就会带来很大的方便。这种方法就称为查表法。

1. 例子：将十进制数据int型转换成十六进制显示出来，要求如果前面有0，不显示。

**private** **static** **void** displayToHex(**int** number)

{

**if**(number == 0) {

System.***out***.println('0');

**return**; //这里return起到了类似break的作用

}

**char**[] disp = {'0','1','2','3','4',

'5','6','7','8',

'9','A','B','C',

'D','E','F'};

**char**[] disp1 = **new** **char**[8];

**int** pos = 8;

**for**(;number != 0;) {

**int** num1 = number & 15;

disp1[--pos] = disp[num1];

number >>>= 4;

}

**for**(**int** i = pos;i<8;i++) {

System.***out***.print(disp1[i]);

}

**return**;//这个return可写可不写，但是上面一个return必须写

}

1. **查表法的应用：**

**private** **static** String getWeek(**int** i)

{

String[] week = {"Error","Monday","Tuesday","Wednesday","Thursday", "Friday" ,"Saturday","Sunday"};

**if**(i<1 || i>7) {

**return** week[0];

}

**return** week[i];

}